Exploration of Polynomial Multiplication Algorithms for Homomorphic Encryption Schemes

Vincent Migliore*, Maria Méndez Real*, Vianney Lapotre*, Arnaud Tisserand‡, Caroline Fontaine†, Guy Gogniat*
*Univ. Bretagne-Sud, UMR CNRS 6285, Lab-STICC, F-56100 Lorient, France, firstname.lastname@univ-ubs.fr
† Institut Mines-Telecom Telecom Bretagne, UMR CNRS 6285, Lab-STICC, Brest, France
‡ CNRS - IRISA - Univ. Rennes 1, Campus ENSSAT, 6 rue Kerampont, 22305 Lannion, France

***Abstract*—Homomorphic encryption schemes allow performing computations in the ciphertext domain, without the need of the secret key. In most promising schemes based on the ring-learning with errors (R-LWE) problem, polynomial multiplication operation is considered an important bottleneck. In this study, a comparison between the Karatsuba and the fast Fourier transform (FFT) multiplication algorithms in the context of homomorphic encryption is proposed in terms of complexity, flexibility and possible optimizations. A complete hardware architecture to speed up polynomial multiplication is provided and impacts of such an architecture on the Karatsuba and the FFT algorithms is thoroughly studied. The study demonstrates that in a realistic architecture, Karatsuba can be a better alternative than the FFT one.**

I. Introduction

Homomorphic encryption schemes are considered as promising in modern cryptography. The main benefit of such encryption schemes is to process data in the cipher domain. More precisely, one can subcontract a service to a third-party server, possibly unreliable, without compromising privacy and data confidentiality. While classical cryptographic schemes have sometimes homomorphic properties, for addition [1] or multiplication [2] operations, it has been necessary to wait until 2009 and C. Gentry [3] breakthrough to discover a scheme able to perform all types of operations with no restriction. Since then, many schemes have been developed in order to optimize practicability of these schemes, the first one needed a public key of 2.25 GB for a ciphertext of 1.5 GB just for one bit of plaintext [4]. Now and since adaptation of homomorphic encryption schemes to the R-LWE problem [5], one can expect optimized and practical parameters, with a public key and a ciphertext close to 360 Kb [6]. However, even if recent schemes may be considered as practical compared to the state of the art, these encryption schemes have still important issues. The main problem is the limitation on consecutive multiplications achievable, due to the existence of an inner noise which is growing until making a decryption procedure impossible. This type of encryption schemes are named somewhat-homomorphic encryption (SHE). In order to break this limitation, a procedure named bootstrapping has been prososed in [3]. It consists on processing inside the cipher domain an operation of decryption and re-encryption, re-initializing the inner noise, and so allowing to have a fully homomorphic encryption (FHE) scheme. Nonetheless this operation has a very important cost, increasing the size of encryption keys and ciphertexts, and thus increasing calculation time, bandwidth usage and storage cost. Consequently, choosing the best scheme for a given problem, the associated protocol and the security level, must be considered as main scientific and technical concerns.

Due to the fact that the bottleneck of R-LWE based encryption schemes is the modular polynomial multiplication operation, various hardware implementations on FPGA are based on the FFT multiplication algorithm [7][8][9][10], which can be used to speed up the key generation step, the homomorphic multiplication step and the decryption step. However, in this work we do not consider any restriction on the polynomial multiplication algorithm, and we demonstrate that Karatsuba's algorithm [11] can be a better alternative compared to the FFT one when dealing with a complete execution scenario.

The main contributions of this work are as follows:

- A thoroughly study of the Karatsuba and the FFT multiplication algorithms for a polynomial multiplication is presented. A comparison in the case of SHE in terms of complexity, flexibility and possible optimizations is also provided.
- A complete and realistic hardware architecture for SHE is introduced in order to evaluate the impact of such an architecture on the Karatsuba and the FFT multiplication algorithms.
- An optimization of the scheduling of input and output coefficients for Karatsuba is proposed in order to reduce the calculation time by proposing pipeline optimizations.
- Key elements for choosing the multiplication algorithm are proposed with respect to hardware resources limitation.

The paper is organized as follows. Section II recaps some key information on SHE cryptosystems based on a R-LWE problem. It presents and compares the Karatsuba and the FFT multiplication algorithms. The proposed hardware architecture and its impact on the two polynomial multiplication algorithms are presented in Section III. Section IV draws some conclusions on the study.

978-1-4673-9406-2/15/$31.00

TABLE I: Practical parameters for SHE schemes [6] based on a R-LWE problem for a security level of 80 bits, where L is the multiplicative depth, and (q, n) are the R-LWE settings

L	$\log_2 q$	n
0	20	512
1	40	1024
3	80	2048
5	128	4096
10	392	8192
50	1225	32768

II. Study of Karatsuba and the FFT Multiplication Algorithms for SHE

A R-LWE problem is constructed in the ideal lattice $\mathbb{Z}_q[X]/\langle f(X)\rangle$, where $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ and $f(X)$ an irreducible polynomial in $\mathbb{Z}_q$ of degree n. The modulus q and the degree of the irreducible polynomial n are chosen in order to satisfy a given level of security and multiplicative depth. Table I recaps some practical parameters for SHE schemes based on [6], considering a security level of 80 bits.

In fast polynomial multiplication algorithms, three steps are required: a pre-computation step, a multiplication step and finally a post-computation step. If one tries to reduce the number of sub-products needed for the multiplication step by choosing a specific algorithm, it will be required to adapt, and possibly use larger pre and post computations. For example, for the basic multiplication algorithm, no pre and post computations are needed, but at a cost of $\mathcal{O}(n^2)$ elementary operations. At the opposite, the FFT multiplication algorithm which requires the smallest number of sub-products, in fact $\mathcal{O}(n \log_2 n)$, has very complex pre and post computations.
For the remainder of the study, we will note with an upper case polynomials of any degree, and with a lower case its coefficients. For a polynomial A, a_i will be the i^{th} coefficient of the associated polynomial. In each polynomial multiplication algorithm, input polynomials will be named A and B, where C will be the computation of $A \times B$:

$$C(x) = \sum_{i=0}^{2n-2} c_i x^i; A(x) = \sum_{i=0}^{n-1} a_i x^i; B(x) = \sum_{i=0}^{n-1} b_i x^i$$

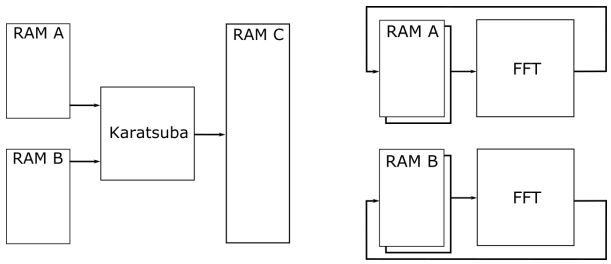
In what follows, a sub-product will refer to an integer multiplication where a sub-multiplication will refers to a polynomial multiplication (smaller than the operands).

A. Basic Multiplication Algorithm

The basic multiplication algorithm is computed by developing the product of two polynomials. In that case, the coefficients of the output polynomial can be calculated by

$$c_k = \sum_{i=0}^{k} a_i b_{k-i}$$

For degree $n-1$ polynomials, the number of elementary operations is $\mathcal{O}(n^2)$. One can notice here that, as stated before, no pre and post computations are needed.



(a) Karatsuba (b) FFT

Fig. 1: High level architectures for Karatsuba and the FFT

B. Karatsuba's Algorithm

Karatsuba's algorithm [11] is an improvement of the basic polynomial multiplication algorithm in order to reduce the number of sub-products in the multiplication step.
First, one needs to split input polynomials A and B of degree $n-1$ into two parts of equivalent size, i.e. $\frac{n}{2}$ coefficients. Let A_H and A_L be two polynomials composed respectively by the coefficients of highest degree of A and lowest degree of A. By the same way, one constructs B_H and B_L. Thus, one therefore obtains $A = A_L + A_H x^{n/2}$ and $B = B_L + B_H x^{n/2}$. When multiplying classically $A \times B$, one obtains :

$$\begin{aligned} A \times B &= (A_L + A_H x^{n/2})(B_L + B_H x^{n/2}) \\ &= A_L B_L + (A_L B_H + A_H B_L) x^{n/2} + A_H B_H x^n \end{aligned}$$

Karatsuba optimization is to notice that the middle factor $(A_L B_H + A_H B_L)$ can be cleverly computed by $(A_L + A_H)(B_L + B_H) - A_L B_L - A_H B_H$. As one can quote, $A_L B_L$ and $A_H B_H$ are already computed and so does not require additionnal operations.
At the end, Karatsuba requires 3 sub-multiplications instead of 4, at a cost of two pre-computations, namely $(A_H + A_L)$ and $(B_H + B_L)$, and two post-computations for the reconstruction of the middle factor. However, these pre and post computations are made of additions and subtractions only. To further reduce the number of sub-products, Karatsuba's algorithm can be recursively applied on the sub-polynomials. For degree $n-1$ polynomials, Karatsuba's complexity is $\mathcal{O}(n^{1.58})$, with a number of sub-products of $3^{\log_2 n}$.

A strategy to implement Karatsuba in hardware consists in designing a small polynomial multiplier of degree p and recursively reconstruct the output polynomial. Because pre and post computations are only composed by additions and subtrations, one can at a first approximation calculates the number of operations required to perform a polynomial multiplication by considering the number of sub-products only. In that case, the number of operations required can be easily calculated by

$$\frac{3^{\log_2 n}}{3^{\log_2 p}} = \frac{3^{\log_2 n}}{3^{\log_2 t/2}},$$

where t is the number of coefficients processed at each operation. Fig. 1a shows an high level architecture for Karatsuba.

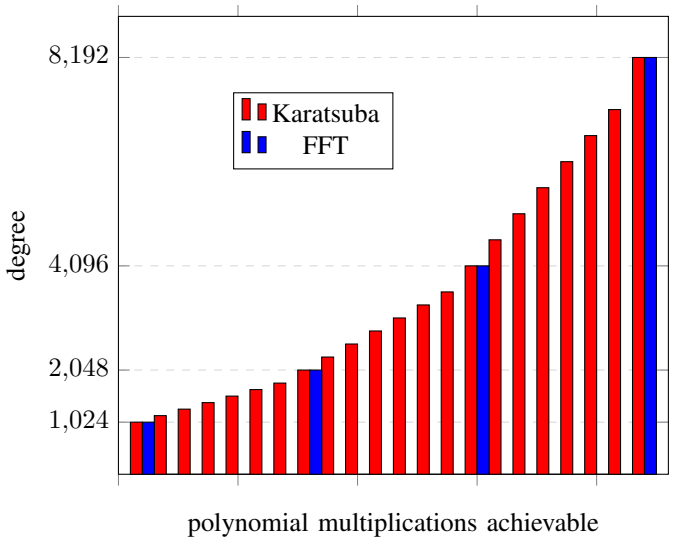


Fig. 2: Polynomial multiplications achievable for Karatsuba and the FFT multiplication algorithms. Karatsuba multiplier has a root polynomial of degree less or equal to 12.

Depending on the degree of the deepest polynomial multiplication, Karatsuba can achieve a polynomial multiplication with various degrees. For example, if the deepest polynomial multiplication is of degree 0 (i.e. integers), Karatsuba can perform a polynomial multiplication of degree 2^{i-1}, where i is the number of times Karatsuba was applied. In general, if the deepest polynomial multiplication is of degree p, Karatsuba can perform a polynomial multiplication of degree $p \times 2^i (p > 0)$. Fig. 2 shows different polynomial multiplications achievable for $p \in [1, 12]$ and $n \in [1024, 8192]$.

C. The FFT Multiplication Algorithm

The FFT multiplication [12] is the algorithm which requires the smallest number of sub-products, but at a cost of complex pre and post computations. The FFT multiplication in the context of R-LWE has been partially studied, we will recall here the main advances so far.

The FFT multiplication algorithm performs a convolution operation between 2 vectors. Applied to 2 polynomials, it performs a polynomial multiplication as it can be seen in Section II-A. Because applying directly a FFT multiplication on two polynomials will led to a wrong convolution, it is necessary to resize the input polynomials. Let $n-1$ be the degree of the input polynomials and k be the smallest integer which satisfy $2^k > 2n-2$. The input polynomials must be filled with 0 in order to be degree 2^k.

To perform the polynomial multiplication, one must first apply a $2n$-FFT operation on each polynomial. Then, a point-wise multiplication is performed on the resulting vectors. The output polynomial is finally reconstructed by applying the inverted FFT algorithm, which is quite similar to the FFT but scaled by n^{-1}. In short, the complete equation is given by:

$$A \times B = \text{IFFT}(\text{FFT}(A) \odot \text{FFT}(B)),$$

Algorithm 1 FFT algorithm

Let ω be a primitive n-th root of unity in $\mathbb{Z}_q$. Let A and A' be a polynomial of a degree less than n, $A(x) = a_0 + \cdots + a_{n-1}x^{n-1}$ and $A'(x) = a'_0 + \cdots + a'_{n-1}x^{n-1}$.

A = reverse_order(A)
for $i = 0$ **to** $\log_2(n) - 1$ **do**
  for $j = 0$ **to** $n/2 - 1$ **do**
    $P_{i,j} = \lfloor \frac{j}{2^{\log_2(n-1-i)}} \rfloor . 2^{\log_2(n-1-i)}$
    $a'_j = a_{2j} + a_{2j+1}\omega^{P_{i,j}} \bmod q$
    $a'_{j+\frac{n}{2}} = a_{2j} - a_{2j+1}\omega^{P_{i,j}} \bmod q$
  end for
  if $i \neq \log_2(n) - 1$ **then**
    $A = A'$
  end if
end for
return A'

where $\odot$ denotes a point-wise multiplication. The FFT algorithm is shown in Algorithm 1. It consists of two nested loop, an inner loop where coefficients of a polynomial A are recombined to produce a polynomial A', and an outer loop which role is to iterate the inner loop $\log_2 d$ times, where d is the size of the FFT. Fig. 1b shows an high level architecture for the FFT polynomial multiplication algorithm. In the followings, the Butterfly unit will refer to the inner loop.

As one can see, pre and post computations are quite complex and require a sub-product in each operation. For a polynomial multiplication of degree $n-1$, the number of sub-products for the FFT is given by

$$\underbrace{2n \log_2 2n}_{\text{FFT}} + \underbrace{2n}_{\text{point-wise mult}} + \underbrace{n log_2 2n}_{\text{IFFT}}.$$

By following the same reasoning than for Karatsuba, one can calculate the number of operations required by neglecting the additions and subtractions. Because the FFT algorithm is a simple nested loop, the number of operations can easily be calculated by $\frac{2n}{t} \log_2(2n)$, where t is the number coefficients processed in parallel. The total number of operations required can now be calculated by

$$\underbrace{\frac{4n}{t} \log_2 2n}_{\text{FFT}} + \underbrace{\frac{4n}{t}}_{\text{point-wise mult}} + \underbrace{\frac{2n}{t} \log_2 2n}_{\text{IFFT}}.$$

Table II compares Karatsuba to the FFT in terms of operations count regarding the number of coefficients processed in parallel. The negative wrapped convolution (NWC) is also compared and will be presented in the next section.

D. Application in the Context of Homomorphic Encryption

As stated before, the R-LWE problem requires a reduction by an irreducble polynomial of degree n. As it can be noticed in Table I, if the irreducible polynomial degree n is a power of two, some specific multiplicative depths L are not reachable and thus can lead to unoptimized parameters for a specific

TABLE II: Comparison of the number of operations required for Karatsuba and the two FFTs multipliers, for polynomials of degree 1023

Coefficients per operation	Number of operations		
	Karatsuba	FFT	NWC
4	19683	17920	8192
8	6561	8960	4096
16	2187	4480	2048
32	729	2240	1024
64	243	1120	512
128	81	560	256
256	27	280	128
512	9	140	64
1024	3	70	32

TABLE III: Summary of pros and cons on Karatsuba and the NWC

	Karatsuba	NWC
multiplications	$\mathcal{O}(n^{1.58})$	$\mathcal{O}(n \log_2 n)$
constraint on n	$\emptyset$	power of 2
constraint on q	$\emptyset$	$q \equiv 1 \pmod{2n}$ and prime
batching technique	yes	no

application. Because Karatsuba can achieve multiplications with degrees of non power of two, parameters of the R-LWE problem can be optimized. At the oposite, FFT polynomial multiplication algorithm which requires such a n may lead to oversize the solution, and so on as n increases. However, at the same time, asymptotic complexity of the FFT compaired to Karatsuba counterbalance this phenomenon.

Karatsuba can also be adapted in order to reduce the complexity of the reduction step required by the R-LWE problem. Nevertheless, in the special case when n is a power of two, the irreducible polynomial can be fixed to $x^n + 1$, and optimizations can be made. First, computation of the reduction is very simple, composed by a simple polynomial addition. Second, one can perform a special form of the FFT, the NWC which requires a n-FFT instead of $2n$. The NWC requires specific parameters to be functional, and in particular n and q must satisfy 4 conditions:

- The polynomial modular must be fixed to $x^n + 1$
- q must be prime
- n must be a power of 2
- $q \equiv 1 \pmod{2n}$

These parameters bring important issues, especially parameter q. Because the integer modular reduction complexity is inversely proportional to the hamming weight of the modulus, finding a modulus close to the modulus required for security purposes, respecting $q \equiv 1 \pmod{2n}$ and with a small hamming weight is a very hard work. Moreover, due to the fact that the modular reduction by q is performed in each pre and post computations, this operation is an important bottleneck.

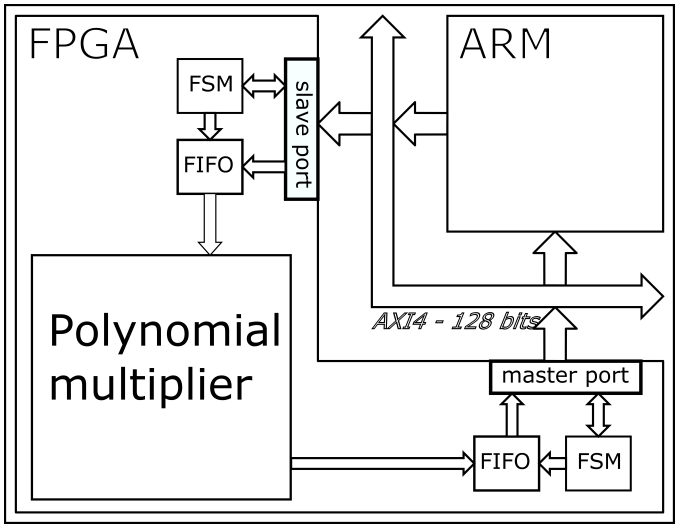


Fig. 3: Proposed architecture for SHE schemes hardware acceleration

In addition, the choice of $x^n + 1$ for the irreducible polynomial can be considered under-optimized. Indeed, because the polynomial $x^n + 1$ is also irreducible in $\mathbb{Z}_2$, the batching technique [13] cannot be used. This technique is very promising for SHE schemes because it "packs" several ciphertexts into a bigger one, where the resulting ciphertext is much smaller than the sum the separated ones. A summary of pros and cons on Karatsuba and the NWC algorithm is shown in Table III.

III. Comparison of the Multiplication Algorithms in a Complete Architecture

A. Proposed Architecture

The proposed architecture is shown in Fig. 3. Due to the complexity of SHE schemes, implementing a complete crypto-processor is a hard work. To accelerate specific steps of homomorphic cryptosystems, an hybrid and flexible structure is needed. This is why the proposed architecture is composed by an ARM which task is to run a SHE software library on a Linux OS, and a FPGA which runs hardware accelerators like a polynomial multiplier. The communication between the ARM and the FPGA is performed through an AXI bus with a 128-bit width.

However, the flexibility of the architecture brings some technological issues. The hardware is penalized by the bandwidth between the ARM and the FPGA. For a polynomial multiplication of degree $n - 1$, the transfer cost can be estimated by $4n \log_2 q / axi\_bus\_width$.

B. Impact of the Architecture on the FFT Algorithm

The FFT algorithm is much impacted by the transfer. Indeed, in each step of the FFT, the entire polynomial is needed in order to complete a Butterfly stage. Thus, the coefficients of the output polynomial cannot be calculated

TABLE IV: Comparaison of Karatsuba, FFT and NWC in terms of operations required, for polynomials of degree $n-1$, in the proposed architecture scenario

n	Karatsuba (basic scheduling)	Karatsuba (optimized scheduling)	FFT	NWC
256	895	767	1074	812
512	1791	1535	2104	1586
1024	3583	3071	4158	3128

during the downloading process. The updated version of the number of operations required is described by

$$\underbrace{2n}_{\substack{\text{transfer in}\\+\\\text{first butterfly}}} \underbrace{+3\left(\log_2(2n)-1\right)\left\lceil\frac{2n}{m}\right\rceil}_{\text{FFT, IFFT}} + \underbrace{\left\lceil\frac{2n}{m}\right\rceil}_{\substack{\text{point-wise}\\\text{mult}}} + \underbrace{2n}_{\substack{\text{transfer out}\\+\\\text{last butterfly}}}$$

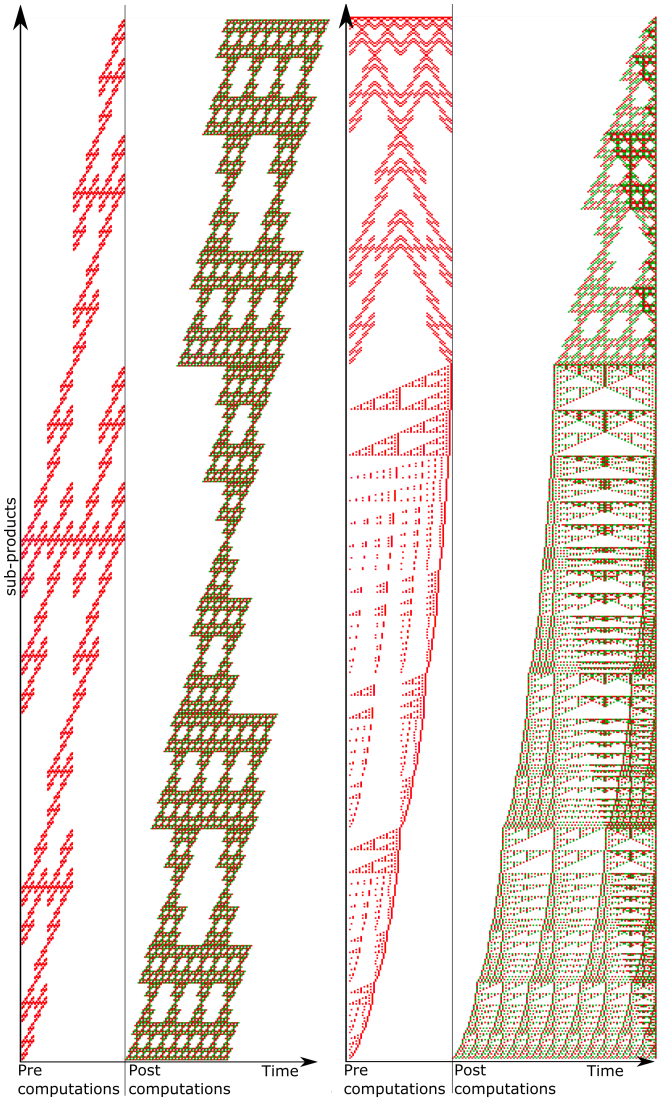
where m is the number of parallel multipliers.
In the case of the NWC, the equation can be reduced to

$$\underbrace{2n}_{\substack{\text{transfer in}\\+\\\text{first butterfly}}} \underbrace{+3\left(\log_2(n)-1\right)\left\lceil\frac{n}{m}\right\rceil}_{\text{FFT, IFFT}} + \underbrace{\left\lceil\frac{n}{m}\right\rceil}_{\substack{\text{point-wise}\\\text{mult}}} + \underbrace{n}_{\substack{\text{transfer out}\\+\\\text{last butterfly}}}$$

C. Impact of the Architecture on Karatsuba's Algorithm

Due to lower data dependencies compared to the FFT, Karatsuba algorithm is able to produce output coefficients even if the pre-computations are not already done. Thus, Karatsuba can pipeline the calculation processes more efficiently than the FFT. Moreover, the scheduling of input and output coefficients can be optimized to further reduce the total calculation time. We investigate different scheduling strategies in order to measure the possibility of speedup. If one carefully examines Karatsuba's algorithm, coefficients with the most important data dependencies are the coefficients closed to the middle, because pre-computations and post-computations are performed in the middle factor. That is why one can optimize the pipelining by sending first the coefficients of low degree and high degree, and finishing by the coefficients in the middle. Due to the penalty of the transfer, we implemented a scheduling algorithm to minimize at the same time the number of operations needed before producing the first output coefficient, and the pre-computations needed between each output coefficient.

Fig. 4 compares the naive sequence with the optimized one. The columns correspond to input coefficients (left) and output coefficients (right). For a given line which corresponds to a sub-product, a red square refers to an addition of the related coefficient, where a green one refers to a subtration. In the optimized strategy, one can observe that output coefficients are gradually ready compared to the naive strategy, and the number of sub-products required between each output coefficients has been reduced. As an optimization, one can exploit the



(a) Naive solution (b) Optimized solution

Fig. 4: Comparison between two scheduling strategies for the input and output coefficients in Karatsuba's algorithm, for a multiplication of polynomials of degree 63

symmetries in pre and post computations to drastically reduce the number of recombinations needed.
Now that one knows how the input and output coefficients are sent, the number of operations can be calculated by an iterative algorithm. Table IV recaps the number of operations needed for Karatsuba and the two FFTs for few values of n, with no hardware restrictions.

D. Key Elements for the Choice of the Multiplication Algorithm

Because low and middle range FPGAs have very limited number of embedded integer multipliers, it is necessary to compare multiplication algorithms with this constraint. Fig. 5 presents the expected speed up of Karatsuba compared to the two FFTs with different number of embedded multipliers. As

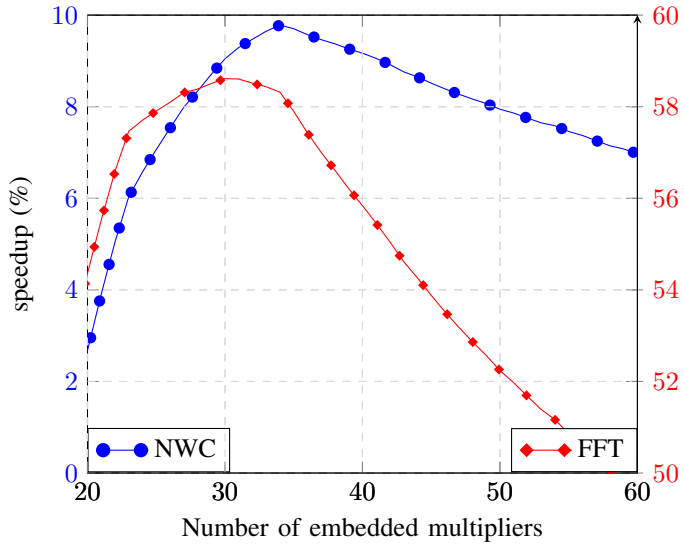


Fig. 5: speedup of Karatsuba's algorithm compared to the FFT and the NWC with respect to the number of embedded multipliers, for n=1024 and 1 coefficient per AXI burst

one can see, Karatsuba can speedup the FFT algorithm by 58% and the NWC by 10% for limited embedded multipliers. When the number of multipliers tends towards zeros, the two FFT algorithms outperform Karatsuba due to the fact that Karatsuba has much more sub-products to achieve. When the number of multipliers tends towards the infinity, the two FFT algorithms improve their calculation time because Butterfly units can be performed fastly and balance the penalty of pre and post computations.

The speedup of Karatsuba compared to the NWC one when one has limited embedded multipliers must also be put into perspective because the polynomial modular reduction remains in the case of Karatsuba, and the optimized scheduling is not fitted to perform the polynomial reduction by $x^n + 1$.

At the end, Karatsuba remains competitive in the case of limited hardware resources, but not in ultra-compact architectures, and when the reduction polynomial is not fixed to $x^n + 1$ because one wants to perform the batching technique.

A deeper study on the complexity of calculating the NWC parameters may be interesting, because as stated before, the bottleneck of the NWC is the integer modular reduction. In that case, Karatsuba may renewed interest compared to the NWC especially because in the NWC implementation in [10], the modulus chosen is under estimated in the case of SHE and the complexity to find such a modulus of higher length has not been presented.

IV. Conclusions And Future Work

A comparison between Karatsuba and the FFT polynomial multiplication was proposed in the context of SHE schemes based on the R-LWE problem, in terms of complexity, flexibility and possible optimizations. A practical architecture for a complete hardware acceleration of SHE schemes has been proposed and impacts on polynomial multiplication algorithms when dealing with this architecture has been studied.

The speedup of Karatsuba compared to the FFT one has been studied for polynomials of few thousands coefficients and for different hardware resources limitations.

Future work will focus on implementing different solutions of the polynomial multiplication in order to have a deeper analysis between each polynomial multiplication algorithm. A study of Karatsuba's algorithm for higher n will also be led, especially in the case of memory limited hardware. Investigations on the way how one can optimize Karatsuba's algorithm in the case of consecutive homomorphic operations will also be led. Toom-Cook algorithm will also be considered for the polynomial multiplication, in particular when algorithms require a division by 3 like an average operation.

A second step of the future work will investigate the acceleration of other primitives of SHE schemes, for example the generation of secret keys which needs the generation of a discrete Gaussian.

Acknowledgment

This study has been financially supported by the french Direction General de l'Armement (DGA).

References

[1] P. Paillier, "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes," in *Proc. of Advances in Cryptology — EUROCRYPT*, ser. NCS, 1999, no. 1592, pp. 223–238.

[2] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-key Cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.

[3] C. Gentry, "A Fully Homomorphic Encryption Scheme," Ph.D. dissertation, Stanford University, 2009.

[4] N. P. Smart and F. Vercauteren, "Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes," in *Proc of Public Key Cryptography – PKC*, ser. LNCS, 2010, no. 6056, pp. 420–443.

[5] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *J. ACM*, vol. 56, no. 6, 2009.

[6] T. Lepoint and M. Naehrig, "A Comparison of the Homomorphic Encryption Schemes FV and YASHE," in *Proc. of AFRICACRYPT*, ser. LNCS, vol. 8469, 2014, pp. 318–335.

[7] N. Göttert, T. Feller, M. Schneider, J. Buchmann, and S. Huss, "On the Design of Hardware Building Blocks for Modern Lattice-Based Encryption Schemes," in *Proc. of Cryptographic Hardware and Embedded Systems – CHES*, ser. LNCS, 2012, no. 7428, pp. 512–529.

[8] T. Pöppelmann and T. Güneysu, "Towards Practical Lattice-Based Public-Key Encryption on Reconfigurable Hardware," in *Proc. of Selected Areas in Cryptography – SAC*, ser. LNCS, 2014, no. 8282, pp. 68–85.

[9] A. Aysu, C. Patterson, and P. Schaumont, "Low-cost and area-efficient FPGA implementations of lattice-based cryptography," in *Proc. of IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, Jun. 2013, pp. 81–86.

[10] D. D. Chen, N. Mentens, F. Vercauteren, S. R. Roy, R. C. Cheung, D. Pao, and I. Verbauwhede, "High-Speed Polynomial Multiplication Architecture for Ring-LWE and SHE Cryptosystems," *IEEE Transactions on Circuits and Systems I*, vol. 62, no. 1, pp. 157–166, Jan. 2015.

[11] A. Karatsuba and Y. Ofman, "Multiplication of multi-digit numbers on automata (in russian)," *Doklady Akad. Nauk SSSR*, vol. 145, no. 2, p. 293–294, 1962, translation in Soviet Physics-Doklady, 44(7), 1963, pp. 595-596.

[12] J. Pollard, "The Fast Fourrier Transform in a Finite Field," in *Mathematics of Computation*, 1971, vol. 25, no. 90, pp. 365–374.

[13] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," in *Proc. of the 3rd Innovations in Theoretical Computer Science Conference*, ser. ITCS, 2012, pp. 309–325.